IN THE CLAIMS:

1. (currently amended) A parallel dispatching and wait signaling method for protecting data items of a dispatcher database of an operating system, the parallel dispatching and wait signaling method comprising the steps of:

creating N local locks, each N local lock for a subset of the dispatcher database, where N > 2, wherein the subsets are formed based upon a natural usage pattern of temporal locality and spatial locality;

acquiring one of the N local locks to perform one of dispatching or wait signaling operation, thereby locking a given subset of the dispatcher database;

limiting access of the data items of the given subset to the one of dispatching or wait signaling operation to be performed for that given subset; and

concurrently maintaining access to data items of unlocked subsets of the dispatcher database so that the operating system maintains ~~a substantially~~ an operational state.

2. (original) The parallel dispatching and wait signaling method of claim 1, wherein:

said acquiring includes acquiring a plurality of the N local locks thereby locking a plurality of subsets of the dispatcher database; and

said limiting includes limiting access of the data items of the locked plurality of subsets to the one of dispatching or wait signaling to be performed.

3. (original) The parallel dispatching and wait signaling method of claim 2, wherein:

said acquiring includes acquiring all N local locks and a global lock thereby locking the dispatcher database; and

said limiting includes limiting access of all data items of the dispatcher database to the one of dispatching or wait signaling to be performed.

4. (original) The parallel dispatching and wait signaling method of claim 1, wherein there are a plurality of one of dispatching or wait signaling operations to be performed and wherein:

acquiring one of the N local locks for each of the plurality of one of dispatching or wait signaling operations to be performed, thereby separately and concurrently locking a plurality of subsets of the dispatcher database for each operation to be performed; and

limiting access of the data items of each locked subset to the one of dispatching or wait signaling operation to be performed for said each locked subset.

5. (currently amended) A method for reducing contention of a highly contended dispatcher lock protecting data items of a dispatcher database of an operating system for a multi-processor computer system, said method comprising the steps of:

~~first determining a methodology to~~ forming one or more dispatch groups including any of threads, resources, and events that frequently interact with each other;

creating N local locks, one for each dispatch group, where $N \geq 2$;

relating each dispatchable object to its dispatch group;

modifying the locking requirements of all code paths of the one or more code paths of the operating system so that the local locks being acquired and released in each of said all code paths are those associated with dispatchable objects touching the code paths;

evaluating the operating system after said modifying the locking requirements so as to determine if the overall performance of the operating system is acceptable based upon user-selected criteria; ~~and~~

when overall performance of the operating system is unacceptable, ~~remodifying the locking requirements of all code paths of the one or more code paths of the operating system until overall performance of the operating system is acceptable~~ counting usage of all the code paths to determine a heaviest path, wherein the heaviest path has a highest usage; and

remodifying the heaviest code path.

6. (original) The method according to claim 5, wherein said relating includes:

separately identifying each dispatch group with a unique identifier; and

identifying each dispatchable object of each group with said unique identifier.

7. (previously presented) A method for providing mutual exclusivity of a dispatcher database of an operating system of a multiprocessor computing system, said method comprising the steps of:

defining a plurality of dispatch groups, each dispatch group being made up of any of threads, resources and events that frequently interact with each other;

defining one or more local locks that protect items making up each dispatch group;

acquiring one of the one or more local locks for any of the threads, resources and events of a given dispatch group that are touched by a code path of ~~the~~ one or more code paths comprising an operating system.

8. (original) A method for reducing contention of a highly contended dispatcher lock protecting data items of a dispatcher database of an operating system for a multi-processor computer system, said method comprising the steps of:

~~first determining a methodology to~~ forming one or more dispatch groups including dispatchable objects that include any of threads, resources and events, wherein the one or more dispatch groups are formed based upon a natural usage pattern;

creating N local locks, one for each dispatch group, where $N \geq 1$;

relating each dispatchable object to its dispatch group;

modifying the locking requirements of each of one or more code paths of the operating system so as to acquire all N local locks and a global lock where the dispatcher lock would have been acquired and so as to release all N local locks and the global lock where the dispatcher lock would have been released;

identifying one code path from the one or more code paths of the operating system; and

optimizing the locking requirements of the identified code path so one or more local locks are acquired and released in the identified code path, the one or more code paths being those associated with the dispatchable objects of the one or more dispatch groups touched by the identified code path.

9. (original) The method according to claim 8, wherein the identified code path includes a plurality of branches, and wherein said optimizing includes optimizing the locking requirements of the identified code path so the one or more locks being acquired

and released in the code path are those associated with the dispatchable objects being touched by each branch of the identified code path.

10. (original) The method according to claim 9, wherein said optimizing includes optimizing the locking requirements of each branch of the identified code path so the one or more locks being acquired and released in each branch are those associated with the dispatchable objects being touched by said each branch.

11. (original) The method according to claim 8, further comprising the step of evaluating the modified operating system after said optimizing the locking requirements so as to determine if the overall performance of dispatching and wait signaling of the operating system is acceptable.

12. (original) The method according to claim 11, wherein in the case where said evaluating determines that the overall performance is not acceptable, then said method includes identifying another code path of the one or more code paths and repeating said steps of optimizing and evaluating for the another identified code path.

13. (original) The method according to claim 12, wherein the code path first identified is the heaviest used code path and wherein the another code path and subsequent code paths are identified sequentially in the direction from the heaviest used code path to a lesser used path.

14. (original) The method according to claim 8, wherein there is one of a plurality or a multiplicity of code paths that access one or more dispatchable objects.

15. (currently amended) A parallel dispatch waiting signaling method for updating a dispatcher database of an operating system for a multiprocessor computing system, the method comprising the steps of:

defining one or more dispatch groups, each dispatch group including dispatchable objects, made up of any of threads, resources or events that are temporally and spatially related;

defining one or more local locks, one for each dispatch group;

relating each dispatchable object to its corresponding dispatch group;

determining if the dispatchable object of an updating operation belongs to a dispatch group;

acquiring one of the one or more local locks, thereby locking a portion of the dispatcher database corresponding to the dispatch group;

updating the dispatcher database portion; and

releasing the local lock following updating.

16. (original) The method according to claim 15, wherein:

said relating includes:

separately identifying each dispatch group with a unique identifier; and

identifying each dispatchable object of each group with said unique identifier; and

said acquiring includes acquiring the one of the local locks for the dispatch group corresponding to the unique identifier.

17. (original) The method according to claim 15, wherein in the case where said determining determines that the dispatchable object of the updating operation does not belong to a dispatch group, then said method further comprises the steps of:

acquiring all locks thereby locking the dispatcher database;

updating the dispatcher database; and

releasing all locks following updating.

18. (original) The method according to claim 17, wherein all locks being acquired includes all local locks and a global lock.

19. (original) The method according to claim 15, wherein while acquiring the one of the one or more local locks, other portions of the dispatcher database are unlocked.

20. (currently amended) A parallel dispatch waiting signaling method for updating data items of a dispatcher database of an operating system for a multiprocessor computing system, the method comprising the steps of:

~~first determining a methodology to~~ forming one or more dispatch groups including dispatchable objects that include any of threads, resources and events from a spatial locality;

creating N local locks, one for each dispatch group, where $N \geq 2$;

relating each dispatchable object to its dispatch group;

identifying one code path from the one or more code paths of the operating system;

optimizing the locking requirements of the identified code path so one or more local locks are acquired and released in the identified code path, the one or more code paths being those associated with the dispatchable objects of the one or more dispatch groups touched by the identified code path; and

determining if the dispatchable object of an updating operation belongs to a dispatch group;

acquiring one of the N local locks, thereby locking a portion of the dispatcher database corresponding to the dispatch group;

updating the dispatcher database portion; and

releasing the local lock following updating.

21. (original) The method according to claim 20, wherein:

said relating includes:

separately identifying each dispatch group with a unique identifier; and

identifying each dispatchable object of each group with said unique identifier; and

the one of the N local locks being acquired corresponds to the lock for the dispatch group corresponding to the unique identifier.

22. (original) The method according to claim 20 further comprising the step of modifying the locking requirements of each of one or more code paths of the operating system that are not optimized so as to acquire all N local locks and a global lock where a

dispatcher lock of the operating system would have been acquired and so as to release all N local locks and the global lock where the dispatcher lock would have been released.

23. (original) The method according to claim 20, wherein in the case where said determining determines that the dispatchable object of the updating operation does not belong to a dispatch group, then said method further comprises the steps of:

acquiring all N locks and a global lock thereby locking the dispatcher database;

updating the dispatcher database; and

releasing all N local locks and the global lock following updating.

24. (original) The method according to claim 20, wherein locking requirements for a plurality of code paths are optimized, and wherein said acquiring, updating and releasing are selectively effected in any one of the plurality of code paths provided that the dispatchable objects to be locked in said any one code path are not locked in any other of the plurality of code paths.

25. (currently amended) An operating system for execution in computer system including a plurality of processors, the operating system including program code, the program code including a sequence of instructions and criteria for protecting and updating data items of a dispatcher database, said sequence of instructions and criteria including

defining one or more dispatch groups, each dispatch group including dispatchable object, made up of any of threads, resources or events that are in a temporal locality;

defining one or more local locks, one for each dispatch group;

relating each dispatchable object to its corresponding dispatch group;

determining if the dispatchable object of an updating operation belongs to a dispatch group;

acquiring one of the one or more local locks to thereby lock a portion of the dispatcher database corresponding to the dispatch group;

updating one or more data items of the locked dispatcher database portion; and

releasing the local lock following updating.

26. (original) The operating system of claim 25, wherein said sequence of instructions and criteria concerning relating each dispatchable object to its corresponding dispatch group includes separately identifying each dispatch group with a unique identifier; and identifying each dispatchable object of each group with said unique identifier; and wherein said sequence of instructions and criteria concerning acquiring one of the one or more local locks includes acquiring the one of the one or more local locks for the dispatch group corresponding to the unique identifier.

27. (original) The operating system of claim 25, wherein in the case where said sequence of instructions and criteria for determining determines that the dispatchable object of the updating operation does not belong to a dispatch group, said program code further includes sequence of instructions and criteria for:

acquiring all locks thereby locking the dispatcher database;

updating the dispatcher database; and

releasing all locks following updating.

28. (original) The operating system of claim 27, wherein all locks being acquired includes all local locks and a global lock.

29. (original) The operating system of claim 25, wherein while acquiring the one of the one or more local locks, other portions of the dispatcher database are unlocked.

30. (currently amended) A computer program product comprising:

a computer-readable medium bearing program code for protecting and updating data items of a dispatcher database of an operating system of a multi-processor computer system, the program code including:

a first computer-readable program code segment for causing the computer system to:

(a) define one or more dispatch groups, each dispatch group including dispatchable objects, made up of any of threads, resources or events that are in a temporal locality and a spatial locality;

(b) define one or more local locks, one for each dispatch group; and

(c) relate each dispatchable object to its corresponding dispatch group.

31. (previously presented) The computer program product of claim 30 wherein the first code segment includes instructions and criteria to:

separately identify each dispatch group with a unique identifier, and

identify each dispatchable object of each group with said unique identifier.

32. (original) The computer program product of claim 30, wherein the program code further includes a second computer-readable program code segment for causing the computer system to (d) determine if the dispatchable object of an updating operation belongs to a dispatch group.

33. (previously presented) The computer program product of claim 32 wherein:

the first code segment includes instructions and criteria to:

separately identify each dispatch group with a unique identifier, and

identify each dispatchable object of each group with said unique identifier; and

the second code segment includes instructions and criteria to determine if the disapatchable object belongs to the dispatch group using the unique identifier.

34. (original) The computer program product of claim 32 further including a third computer-readable code segment for causing the computer system to (e) acquire one of the one or more local locks to thereby lock a portion of the dispatcher database corresponding to the dispatch group.

35. (previously presented) The computer program product of claim 34, wherein the third code segment includes instructions and criteria that, while acquiring the one of the one or more local locks, other portions of the dispatcher database are to be unlocked.

36. (original) The computer program product of claim 34 further including a fourth computer-readable code segment for causing the computer system to:

(f) update one or more data items of the locked dispatcher database portion; and

(g) release the acquired local lock following updating.

37. (previously presented) The computer program product of claim 36, wherein when it is determined by the second code segment that the dispatchable object of the updating operation does not belong to a dispatch group, said third and fourth code segments further includes instructions and criteria for:

acquiring all locks thereby locking the dispatcher database;

updating the dispatcher database; and

releasing all locks following updating.

38. (original) The method according to claim 37, wherein the all locks being acquired includes all local locks and a global lock.

39. (previously presented) The computer program product of claim 34 wherein:

the first code segment includes instructions and criteria to:

separately identify each dispatch group with a unique identifier, and

identify each dispatchable object of each group with said unique identifier; and

the third code segment includes instructions and criteria to acquire the one of the one or more local locks for the dispatch group corresponding to the unique identifier.

40. (currently amended) A multiprocessor computer system comprising:

a plurality of processors;

a physical memory accessed and used by the plurality of processors;

program code for execution within the plurality of processors; and

wherein the program code comprises criteria and a sequence of instructions to protect and update data items of a dispatcher database, said instructions and criteria including:

defining one or more dispatch groups, each dispatch group including dispatchable object, made up of any of threads, resources or events that are in a temporal locality and a spatial locality;

defining one or more local locks, one for each dispatch group;

relating each dispatchable object to its corresponding dispatch group;

determining if the dispatchable object of an updating operation belongs to a dispatch group;

acquiring one of the one or more local locks to thereby lock a portion of the dispatcher database corresponding to the dispatch group;

updating one or more data items of the locked dispatcher database portion; and

releasing the local lock following updating.

41. (original) The multiprocessor computer system of claim 40, wherein the program code criteria and sequence of instructions concerning relating each dispatchable object to its corresponding dispatch group includes separately identifying each dispatch group with a unique identifier; and identifying each dispatchable object of each group with

said unique identifier; and wherein the instructions and criteria concerning acquiring one of the one or more local locks includes acquiring the one of the one or more local locks for the dispatch group corresponding to the unique identifier.

42. (original) The multiprocessor computer system of claim 40, wherein in the case where the program code criteria and sequence of instructions for determining determines that the dispatchable object of the updating operation does not belong to a dispatch group, the program code criteria and sequence of instructions includes:

acquiring all locks thereby locking the dispatcher database;

updating the dispatcher database; and

releasing all locks following updating.

43. (previously presented) The method according to claim 20, wherein while the one of the N local locks is acquired, other portions of the dispatcher database are unlocked.